

Mendelova zemědělská a lesnická univerzita v Brně  
Provozně ekonomická fakulta

---

# **Dokumentace překladače JUGA**

**Projekt do předmětu Teorie programovacích jazyků**

Vypracovali:

Bc. Jůva Jan

Bc. Gábriš Pavel

Brno 2008

## Obsah

<b>1</b>	<b>Popis jazyka</b>	<b>3</b>
<b>2</b>	<b>Lexikální prvky jazyka</b>	<b>3</b>
<b>3</b>	<b>Lexikální analýza</b>	<b>5</b>
3.1	Gramatiky pro lexikální prvky . . . . .	5
3.2	Regularizace . . . . .	7
3.3	Jazyk interpretu . . . . .	7
3.4	Stanovení významnosti lexikálních jednotek . . . . .	8
3.5	Sestavení gramatiky lexikálního analyzátoru . . . . .	8
<b>4</b>	<b>Syntaktický analyzátor</b>	<b>10</b>
4.1	Ověření LL (1) . . . . .	11
	<b>Přílohy</b>	<b>12</b>
<b>A</b>	<b>Testovací soubory</b>	<b>13</b>
A.1	Malá násobilka . . . . .	13
A.2	Faktoriál . . . . .	14
A.3	Pyramida . . . . .	14
A.4	Hanojské věže . . . . .	15

## 1 Popis jayzka

Progarmovcí jayzk JUGA je navržen tak, aby byl schopen podporovat znaky a řetězce, čísla celá a čísla desetinná, přiřazovací příkazy, identifikátory proměnných, aritmetické a logické výrazy, rozhodovací příkazy dále programové cykly, uživatelské funkce a vstupně-výstupní funkce. V neposlední řadě jazyk disponuje doplňujícími protředky jako jsou oddělovače příkazů, programové bloky, tzv. bílé znaky a dva typy komentářů.

## 2 Lexikální prvky jazyka

Zde uvádíme výčet jednotlivých prvků jazyka JUGA.

### Čísla

Jazyk podporuje kladná i záporná čísla a to jak v celočíselné tak i desetinné formě. Není podporována jiná než desítková soustava. Oddělovačem desetinné části je čárka.

```
+99,99
```

### Identifikátory proměnných

Proměnné jsou uvozeny znakem dolaru. Název proměnné musí vždy začínat tímto znakem, přičemž musí následovat alespoň jeden další znak a sice číslo 0..9 nebo libovolné písmeno a znak podtržítka.

```
$prom_098
```

### Operátory

Podporujeme standardní aritmetické operátory, pro zbytek po celočíselném dělení slouží operátor %. Rovnost se zapisuje pomocí znaku =, nerovnost pomocí znaků !=. Pro operátor zřetězení je použito +. Pro logické AND je použit znak &, OR nahrazuje znak |.

```
$prom * 3 % 2 <> 1
```

### Řetězce

Řetězce se uzavírají do uvozovek "" a mohou obsahovat libovolné znaky z abecedy interpretu. Mohou se použít i prázdné či jednoznakové řetězce.

```
"ahoj!"
```

## Oddělovače

Mezi oddělovače patří znak mezery, tabulátoru a enter.

## Komentáře

Jsou podporovány nejen jednořádkové ale i víceřádkové komentáře. Řádkový komentář začíná dvěma znaky `//` a je ukončen koncem řádku. Komentář víceřádkový začíná znaky `/*` a končí znaky `*/`.

```
// Toto je jednořádkový komentář
/* Toto je víceřádkový komentář */
```

## Standardní funkce

Funkce pro načtení ze vstupu začíná klíčovým slovem `READ`. Funkce `WRITE` slouží pro výpis.

```
READ($cislo);
WRITE("Bylo zadáno číslo: " + $cislo);
```

## Příkaz přiřazení

Příkaz přiřazení je dán dvěma znaky `<:`.

```
$promena <: hodnota;
```

### 3 Lexikální analýza

#### 3.1 Gramatiky pro lexikální prvky

##### Gramatika jazyka čísel (D)

$$G_D = (N_D, \Sigma_D, P_D, S_D)$$

$$N_D = \{S_D, D_1, D_2\}$$

$$\Sigma_D = \{c, +, -, , \} \quad c \sim 0..9$$

$$P_D :$$

$$S_D \rightarrow +D_1 \mid -D_1 \mid D_1$$

$$D_1 \rightarrow cD_1 \mid c \mid c, D_2$$

$$D_2 \rightarrow cD_2 \mid c$$

##### Gramatika jazyka operátorů (O)

$$G_O = (N_O, \Sigma_O, P_O, S_O)$$

$$N_O = \{S_O\}$$

$$\Sigma_O = \{+, -, *, /, \&, |, !, (, ), <, >, =, ^, \% \}$$

$$P_O :$$

$$S_O \rightarrow + \mid - \mid * \mid / \mid \& \mid | \mid ! \mid ( \mid ) \mid < \mid > \mid <= \mid >= \mid = \mid <> \mid ^ \mid \%$$

##### Gramatika jazyka přiřazení (A)

$$G_A = (N_A, \Sigma_A, P_A, S_A)$$

$$N_A = \{S_A\}$$

$$\Sigma_A = \{<, :\}$$

$$P_A :$$

$$S_A \rightarrow <: :$$

##### Gramatika jazyka řetězců (S)

$$G_S = (N_S, \Sigma_S, P_S, S_S)$$

$$N_S = \{S_S, S_1, S_2\}$$

$$\Sigma_S = \{", \backslash, \Delta\}$$

$$P_S :$$

$$S_S \rightarrow "S_1$$

$$S_1 \rightarrow \Delta S_1 \mid " \mid \backslash S_2$$

$$S_2 \rightarrow \triangle S_1 \mid "S_1 \mid \backslash S_1$$

$$\triangle = \Sigma - \{", \backslash\}$$

### Gramatika jazyka proměnných (V)

$$G_V = (N_V, \Sigma_V, P_V, S_V)$$

$$N_V = \{S_V, V_1, V_2\}$$

$$\Sigma_V = \{p, c, -, \$\} \quad p \sim A..Z, a..z \quad c \sim 0..9$$

$$P_V :$$

$$S_V \rightarrow \$V_1$$

$$V_1 \rightarrow pV_2 \mid cV_2 \mid -V_2$$

$$V_2 \rightarrow pV_2 \mid cV_2 \mid -V_2$$

### Gramatika klíčových slov (R)

$$G_R = (N_R, \Sigma_R, P_R, S_R)$$

$$N_R = \{S_R, R_1\}$$

$$\Sigma_R = \{p, c, -\}$$

$$P_R :$$

$$S_R \rightarrow pR_1 \mid p$$

$$R_1 \rightarrow pR_1 \mid cR_1 \mid -R_1 \mid p \mid c$$

### Gramatika jazyka příkazových bloků (B)

$$G_B = (N_B, \Sigma_B, P_B, S_B)$$

$$N_B = \{S_B\}$$

$$\Sigma_B = \{\{, \}\}$$

$$P_B :$$

$$S_B \rightarrow \{ \mid \}$$

### Gramatika ukončovacího znaku příkazu (T)

$$G_T = (N_T, \Sigma_T, P_T, S_T)$$

$$N_T = \{S_T\}$$

$$\Sigma_T = \{;\}$$

$$P_T :$$

$$S_T \rightarrow ;$$

**Gramatika oddělovače parametrů funkcí (P)**

$$G_P = (N_P, \Sigma_P, P_P, S_P)$$

$$N_P = \{S_P\}$$

$$\Sigma_P = \{, \}$$

$$P_P :$$

$$S_P \rightarrow ,$$

**Gramatika jazyka bílých znaků (W)**

$$G_W = (N_W, \Sigma_W, P_W, S_W)$$

$$N_W = \{S_W\}$$

$$\Sigma_W = \{\_, TAB, \leftrightarrow\}$$

$$P_W :$$

$$S_W \rightarrow \_ \mid TAB \mid \leftrightarrow$$

**Gramatika jazyka komentářů (C)**

$$G_C = (N_C, \Sigma_C, P_C, S_C)$$

$$N_C = \{S_C, C_1, C_2, C_3\}$$

$$\Sigma_C = \{/, *\}$$

$$P_C :$$

$$S_C \rightarrow //C_1 \mid / * C_2$$

$$C_1 \rightarrow \blacksquare C_1 \mid /C_1 \mid * C_1 \mid \leftrightarrow$$

$$C_2 \rightarrow \square C_2 \mid \leftrightarrow C_2 \mid * C_3 \mid /C_2$$

$$C_3 \rightarrow / \mid * C_3 \mid \square C_2$$

$$\blacksquare = \Sigma - \{\leftrightarrow\}$$

$$\square = \Sigma - \{*/\}$$

**3.2 Regularizace****3.3 Jazyk interpretu**

$$\Sigma = \{A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, \\ a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \\ +, -, ., *, \&, /, \backslash, |, !, (, ), <, >, =, :, ", \{, \}, \_, TAB, \leftrightarrow, ;, [, ], \neg, \$, ^, \%, \#, ,, ?, @, \sim, ', '\}$$

### 3.4 Stanovení významnosti lexikálních jednotek

#### Nevýznamové jednotky ( $S_{od}$ )

$$S_{od} \rightarrow //C_1 \mid / * C_2 \mid \neg S_{od} \mid TABS_{od} \mid \leftrightarrow S_{od} \mid \epsilon$$

$$C_1 \rightarrow \blacksquare C_1 \mid \leftrightarrow S_{od}$$

$$C_2 \rightarrow \square C_2 \mid * C_3$$

$$C_3 \rightarrow \square C_2 \mid /S_{od}$$

$$\blacksquare = \Sigma - \{\leftrightarrow\}$$

$$\square = \Sigma - \{*/\}$$

#### Významové jednotky ( $S_{vt}$ )

$$L_{vt} = D \cup O \cup A \cup S \cup V \cup R \cup B \cup T$$

$$S_{vt} \rightarrow S_D \mid S_O \mid S_A \mid S_S \mid S_V \mid S_R \mid S_B \mid S_T$$

$$S_{vt} \rightarrow +D_1 \mid -D_1 \mid D_1 \mid + \mid - \mid * \mid / \mid \& \mid | \mid ! \mid ( \mid ) \mid < \mid > \mid <= \mid >= \\ \mid = \mid <> \mid ^ \mid \% \mid <: \mid "S_1 \mid \$V_1 \mid pR_1 \mid p \mid \{ \mid \} \mid ; \mid ,$$

### 3.5 Sestavení gramatiky lexikálního analyzátoru

#### Lineární lexikální analyzátor

Platí:  $lex \sim L_{od} * L_{vt}$

$$S \rightarrow //C_1 \mid / * C_2 \mid \neg S \mid TABS \mid \leftrightarrow S \mid S_{vt} \mid \epsilon$$

$$D_1 \rightarrow cD_1 \mid c \mid c.D_2$$

$$D_2 \rightarrow cD_2 \mid c$$

$$S_1 \rightarrow \triangle S_1 \mid " \mid \backslash S_2$$

$$S_2 \rightarrow \triangle S_1 \mid "S_1 \mid \backslash S_1$$

$$V_1 \rightarrow pV_2 \mid cV_2 \mid \neg V_2$$

$$V_2 \rightarrow pV_2 \mid cV_2 \mid \neg V_2$$

$$R_1 \rightarrow pR_1 \mid cR_1 \mid \neg R_1 \mid p \mid c$$

$$C_1 \rightarrow \blacksquare C_1 \mid \leftrightarrow S$$

$$C_2 \rightarrow \square C_2 \mid * C_3$$

$$C_3 \rightarrow \square C_2 \mid /S$$

$$S_{vt} \rightarrow +D_1 \mid -D_1 \mid D_1 \mid + \mid - \mid * \mid / \mid \& \mid | \mid ! \mid ( \mid ) \mid < \mid > \mid <= \mid >= \\ \mid = \mid <> \mid ^ \mid \% \mid <: \mid "S_1 \mid \$V_1 \mid pR_1 \mid p \mid \{ \mid \} \mid ; \mid ,$$

$$\triangle = \Sigma - \{", \backslash\}$$

$$\blacksquare = \Sigma - \{\leftrightarrow\}$$

$$\square = \Sigma - \{*/\}$$



### Regularizovaný lexikální analyzátor

Po provedení regularizace získáváme následující regulární gramatiku lexikálního analyzátoru:

$$S \rightarrow [C_4 \mid \_S \mid TABS \mid \leftarrow S \mid + D_1 \mid - D_1 \mid cD_1 \mid c \mid cD_3 \mid + \mid - \mid * \mid / \mid \&O_1 \mid |O_2 \mid ! \mid ( \mid ) \mid < \mid > \mid < O_3 \mid > O_3 \mid = \mid < O_4 \mid ^ \mid \% \mid : O_3 \mid "S_1 \mid \$V_1 \mid pR_1 \mid p \mid \{ \mid \} \mid ; \mid , \mid \epsilon$$

$$D_1 \rightarrow cD_1 \mid c \mid cD_3$$

$$D_2 \rightarrow cD_2 \mid c$$

$$S_1 \rightarrow \bullet S_1 \mid " \mid \backslash S_2 \mid \leftarrow S_1 \mid * S_1$$

$$S_2 \rightarrow \bullet S_1 \mid "S_1 \mid \backslash S_1 \mid \leftarrow S_1 \mid /S_1$$

$$V_1 \rightarrow pV_2 \mid cV_2 \mid \_V_2$$

$$V_2 \rightarrow pV_2 \mid cV_2 \mid \_V_2$$

$$R_1 \rightarrow pR_1 \mid cR_1 \mid \_R_1 \mid p \mid c$$

$$C_1 \rightarrow \bullet C_1 \mid "C_1 \mid \backslash C_1 \mid * C_1 \mid \leftarrow S$$

$$C_2 \rightarrow \bullet C_2 \mid "C_2 \mid \backslash C_2 \mid * C_3 \mid \leftarrow C_2$$

$$C_3 \rightarrow \bullet C_2 \mid "C_2 \mid \backslash C_2 \mid /S \mid \leftarrow C_2$$

$$S_{vt} \rightarrow +D_1 \mid -D_1 \mid cD_1 \mid c \mid cD_3 \mid + \mid - \mid * \mid / \mid \&O_1 \mid |O_2 \mid ! \mid ( \mid ) \mid < \mid > \mid < O_3 \mid > O_3 \mid = \mid < O_4 \mid ^ \mid \% \mid : O_3 \mid "S_1 \mid \$V_1 \mid pR_1 \mid p \mid \{ \mid \} \mid ; \mid ,$$

$$C_4 \rightarrow > C_1 \mid /C_2$$

$$D_3 \rightarrow .D_2$$

$$O_1 \rightarrow \&$$

$$O_2 \rightarrow |$$

$$O_3 \rightarrow =$$

$$O_4 \rightarrow >$$

$$\bullet = \Sigma - \{", \backslash, \leftarrow, /\}$$

## 4 Syntaktický analyzátor

Gramatika je upravená tak že, neobsahuje bezprostřední levou rekurzi a bylo možné určit množiny First a Follow.

$PROGRAM \rightarrow BLOCK$   
 $BLOCK \rightarrow \{ \text{BRANCH} \}$   
 $BRANCH \rightarrow \{ \text{STATEMENT}; \text{BRANCH} \}$   
 $FNCDCE \rightarrow @IDENTIFIKATOR ( L )$   
 $L \rightarrow \text{prom} \$IDENTIFIKATOR ; L \mid \text{cte} \$IDENTIFIKATOR ; L$   
 $GET\_F \rightarrow ( S )$   
 $STATEMENT \rightarrow \text{VETVENI} \mid \text{CYKLUS} \mid \text{VYSTUP} \mid \text{VSTUP} \mid \text{GET} \mid \text{GET\_F}$   
 $\text{VETVENI} \rightarrow \text{iff} (A) \text{ BLOCK TEHEN BLOCK ffi}$   
 $\text{CYKLUS} \rightarrow \text{untill} (A) \text{ BLOCK FINIS}$   
 $\text{VSTUP} \rightarrow \text{read} (\$IDENTIFIKATOR)$   
 $\text{VYSTUP} \rightarrow \text{write}(A)$   
 $\text{GET} \rightarrow \$IDENTIFIKATOR <: A$   
 $A \rightarrow B\bar{A} \mid \epsilon$   
 $\bar{A} \rightarrow O\bar{A}$   
 $B \rightarrow !C \mid C$   
 $O \rightarrow |B| \&b$   
 $C \rightarrow D \text{ BRANCH}$   
 $\bar{C} \rightarrow PD$   
 $P \rightarrow < \mid > \mid = \mid \leq \mid \geq \mid !=$   
 $D \rightarrow E\bar{D}$   
 $\bar{D} \rightarrow +E \mid -E \mid \epsilon$   
 $E \rightarrow F\bar{E}$   
 $\bar{E} \rightarrow *F \mid /F \mid \%F$   
 $F \rightarrow (A) \mid \text{cislo} \mid \text{retezec} \mid \$identifikator \mid \#identifikator$

## 4.1 Ověření LL (1)

$FF(PROGAM \rightarrow BLOCK) = \{ \{ \}$   
 $FF(BLOCK \rightarrow \{BRANCH\}) = \{ \{ \}$   
 $FF(BRANCH \rightarrow STATEMENT; BRANCH) =$   
 $FI(STATEMENT) = \{ iff, until, write, read, \$identifikator, @identifikator \}$   
 $FI(BRANCH) = \{ \{ \}$   
 $FF(FNCDEC \rightarrow @identifikator(L)) = \{ @identifikator \}$   
 $FF(L \rightarrow prom\$identifikator; L \mid cte\$identifikator; L) = \{ prom, cte \}$   
 $FF(GET\_F \rightarrow (S)) = \{ ( \}$   
 $FF(S \rightarrow SA) = \{ (, cislo, retzec, \$identifikator, \#identifikator \}$   
 $FF(STATEMENT \rightarrow VETVENI \mid CYKLUS \mid VYSTUP \mid VSTUP \mid GET \mid GET\_F) =$   
 $\{ iff, until, write, read, \$identifikator, @identifikator \}$   
 $FF(VETVENI \rightarrow iff (A) BLOCK TEHEN BLOCK ffi) = \{ iff \}$   
 $FF(CYKLUS \rightarrow untill (A) BLOCK FINIS) = \{ until \}$   
 $FF(VSTUP \rightarrow read (\$IDENTIFIKATOR)) = \{ read \}$   
 $FF(VYSTUP \rightarrow write(A)) = \{ write \}$   
 $FF(GET \rightarrow \$IDENTIFIKATOR <: A) = \{ \$identifikator \}$   
 $FF(A \rightarrow B\bar{A} \mid \epsilon) =$   
 $FI(A \rightarrow B\bar{A}) = \{ ! \}$   
 $FI(A \rightarrow \epsilon) = FO(A) = \{ ) \}$   
 $FF(\overline{A} \rightarrow \overline{OA}) = \{ |, \& \}$   
 $FF(B \rightarrow !C \mid C) =$   
 $FI(B \rightarrow !C) = \{ ! \}$   
 $FI(B \rightarrow !C) = \{ (, cislo, retzec, \$identifikator, \#identifikator \}$   
 $FF(O \rightarrow |B \mid \&b) = \{ |, \& \}$   
 $FF(C \rightarrow D BRANCH) = \{ (, cislo, retzec, \$identifikator, \#identifikator \}$   
 $FF(\bar{C} \rightarrow PD) = \{ <, >, =, \leq, \geq, != \}$   
 $FF(P \rightarrow < \mid > \mid = \mid \leq \mid \geq \mid !=) = \{ <, >, =, \leq, \geq, != \}$   
 $FF(D \rightarrow E\bar{D}) = \{ (, cislo, retzec, \$identifikator, \#identifikator \}$   
 $FF(\bar{D} \rightarrow +E \mid -E \mid \epsilon) =$   
 $FI(\bar{D} \rightarrow +E \mid -E) = \{ +, - \}$   
 $FI(\bar{D} \rightarrow \epsilon) = \{ <, >, =, \leq, \geq, != \}$   
 $FF(\bar{E} \rightarrow *F \mid /F \mid \%F) = \{ *, /, \% \}$   
 $FF(F \rightarrow (A) \mid cislo \mid retzec \mid \$identifikator \mid \#identifikator) = \{ (, cislo, retzec,$   
 $\$identifikator, \#identifikator \}$

## **Přílohy**

## A Testovací soubory

Příloha obsahuje zdrojové kódy čtyř ukázkových programů.

### A.1 Malá násobilka

Program na základě zadaného čísla vypíše malou násobilku.

```
{  
WRITE("Zadejte cislo:");  
READ($cis);  
$pom <: 1;  
  
WRITE(":: Probiha nasobeni ::");  
UNTIL($pom <= 10){  
  //WRITE($cis);  
  //WRITE("*");  
  //WRITE($pom);  
  //WRITE("=");  
  $vysledek <: $cis * $pom;  
  WRITE($vysledek);  
  $pom <: $pom + 1;  
  
} FINISH;  
  
}EXIT
```

## A.2 Faktoriál

Program vypočítá faktoriál zadaného čísla.

```
{  
WRITE("Zadejte cislo, pro ktere se ma vypocitat faktorial:");  
READ($cislo);  
  
$pom <: 1;  
$naval <: 1;  
  
UNTIL($cislo > 1){  
$pom <: $pom * $cislo;  
$cislo <: $cislo - 1;  
$naval <: $pom;  
  
} FINISH;  
WRITE("Vysledek je:");  
WRITE($naval);  
  
}EXIT
```

## A.3 Pyramida

```
{  
// vypis pyramidy z libovolnych znaku  
  
WRITE("Zadejte pozadovany pocet radku:");  
READ($pocet);  
WRITE("z jakeho znaku:");  
READ($znak);  
  
$radek <: 1;  
  
UNTIL($radek <= $pocet){  
$pom <: $pocet - $radek;  
$pocni <: $radek - 1;  
  
UNTIL($pom > 0){  
WRITE(" ");  
$pom <: $pom - 1;  
}FINISH;  
}
```

```
IFF($radek > 1){
    $pom <: $radek - 1;
    UNTIL($pom > 0){
        WRITE(" ");
        WRITE($znak);
        $pom <: $pom - 1;
    }FINISH;
}FFI;

WRITE(" ");
$radek <: $radek + 1;
}FINISH;

}EXIT
```

## A.4 Hanojské věže

```
@preskladat($vyska,$od,$pres,$kam){

    IFF($vyska > 0){

        #preskladat($vyska - 1,$od,$kam,$pres);
        WRITE($od);
        WRITE(">>>");
        WRITE($kam);
        WRITE("\n");
        #preskladat($vyska - 1,$pres,$od,$kam);

    }FFI;
}

{
    WRITE("Zadej vysku veze:");
    READ($pocet);

    #preskladat($pocet;1,2,3);

}EXIT
```